METHOD OF EXTERNALIZING RELATIONAL AND ASN.1-FORMATTED DATA INTO XML FORMAT

Field of the Invention

This invention relates to the field of conversion of data from one format to another with a digital data processing device or devices. More particularly, the invention relates to a process of transforming legacy data in a legacy database into a Extensible Markup Language ("XML") format which is more useful for current applications.

Background of the Invention

A forerunner of XML, HTML ("Hyptertext Markup Language") was conceived as an easily understandable language for the exchange of scientific and other technical documents. HTML addressed the problem of SGML ("Standard Generalized Markup Language") complexity by specifying a small set of structural and semantic tags suitable for authoring relatively simple documents. In addition to simplifying the document structure, HTML added support for hypertext. Multimedia capabilities were added later.

In a brief period of time, HTML became very popular and quickly outgrew its original purpose. Since HTML's inception, many new elements have been devised for use within HTML (as a standard) and for adapting HTML to vertical, highly specialized, markets.

A severe shortcoming of HTML is its lack of data structuring mechanisms. HTML documents capture presentation and rendering aspects of marked-up documents, but the formalism does not lend itself to describe the structure of data which is exchanged between computing entities in typical client-server applications.

XML is a proposed standard for describing the structure of semi-structured data. The formalism supports the description of concrete mark-up languages which allow the specification of hierarchical (i.e. tree-like) data structures. Concrete mark-up languages can be specially adapted to particular application domains, such as the airline industry, finance industry, etc. where said concrete mark-up languages allow to model data entities per-use by these applications. XML can also be used to specify HTML as a concrete mark-up language. More information about XML can be found in "Extensible Markup Language (XML) 1.0: W3C Recommendation 10 February 1998", http://www.w3.org/TR/REC-xml. and E.H. Harold, XML Extensible Markup Language, (IDG Books 1998).

XML, proposed by W3C (the World-Wide Web standardization body), has found wide-spread acceptance in the industry and is rapidly becoming the *lingua franca* for data representation/description mechanisms used throughout the World Wide Web. It is an open specification and several major industry leaders, among them Microsoft and IBM, are pushing for the use of XML formatted data exchanged between the various IT systems and sub-systems which make up an enterprise as well as a personal computing environment.

Abstract Syntax Notation ("ASN", currently version One, "ASN.1") serves a similar purpose: data structures can be described abstractly using the ASN.1 syntax. Its initial intention was to provide a scheme to specify the structure of data to be exchanged between computer systems in a system-independent, common representation. Therefore, ASN.1 also provides several concrete standardized transfer encodings such as "Basic Encoding Rules" (BER), "Definite Encoding Rules" (DER) etc.. Due to the complexity of the ASN.1 data description language as well as the multiplicity of encoding rules, ASN.1 usage has been restricted to a limited set of IT applications, mainly related to IT security (e.g., directories, public key infrastructure).

Most data on the Internet is stored in legacy databases, many of which are using a relational data base model and in some cases ASN.1 encoding. For both cases, it is of commercial value to externalize such data into an XML compliant format in order to enable new Internet based applications.

For relational databases as well as for ASN.1 data, some work has been done to map actual data into XML compliant data formats. However, these transformations do not generate the meta-data description, embodied by an XML Document Type Definition (DTD). For example, one proposed translation for ASN.1 encoded data, published at http://asf.gils.net/xer/standard.html, only defines a mapping from one concrete ASN.1 data module onto one concrete XML data representation without generating the XML DTD.

What is needed, therefore, is an efficient method to externalize legacy data into an XML compliant format where the format is specified by an automatically generated XML meta-description (i.e. DTD). Such procedures will enable the access and processing of legacy data by new, Internet generation applications.

Summary of the Invention

A computer-readable medium is encoded with a method of externalizing legacy data from a legacy database on a data resource into an Extensible Markup Language ("XML") -compliant format where the format is specified by an automatically generated XML meta-description, thus enabling access and processing of legacy data by XML-compliant applications. The method includes an exploration/adaptation step for exploration of the data resource, and a production step for generation of the XML data out of the legacy data. The method automates the data format mapping between legacy databases and an XML compliant representation of that data.

As another feature of the invention, the method allows conversion between ASN.1 specification and an XML meta data description or Document Type Description ("DTD") and the transformation of ASN.1 encoded data into XML compliant format adhering to said XML DTD.

In an advantage of the invention, accessing the relational data base model through an XML compliant data format allows companies such as banks, insurance companies, transportation enterprises, and department stores to open their databases to customers.

Any relational database can be accessed and its contents converted into a XML representation that is much more accessible to Internet applications, thus facilitating e-commerce.

Brief Description of the Drawings

Fig. **1** is a flow diagram of the method of converting legacy data into an XML compliant data format.

Fig. **2** is a flow diagram of a submethod that converts ASN.1 data into XML compliant data.

Detailed Description of the Preferred Embodiment

Referring now to FIG. **1** , the method 10 of the present invention is a method for externalizing legacy data from a legacy database on a data resource into an Extensible Markup Language ("XML") -compliant format where the format is specified by an automatically generated XML meta-description. This reformatting enables access and processing of legacy data by XML-compliant applications.

The method 10 automates the data format mapping between relational databases or ANS.X formatted databases and an XML compliant representation of that data, by performing the following steps: (1) an exploration/adaptation step 12 for exploration of the data resource, and (2) a mapping step 14 for generation of the XML data out of the legacy data.

The submethod 14a or 14b to be applied in the second mapping step 14 is determined by the data format of the legacy data. If the data is in relational data format, the submethod 14a is applied, and includes three substeps. In a first substep 16, relations (i.e. tables) within the relational data model are mapped onto XML elements. In a second substep 20, tuples (i.e., rows) within a relation are mapped onto XML elements which are nested within the XML "table" element; and

in a third substep 22, attributes (i.e., columns) of tuples are mapped onto XML elements which are nested within the XML "row" element.

As a concrete example, consider relation T with attributes (a1, a2, ..., an). This can be represented using the following XML data description:

```
<!ELEMENT TABLE_T(ROW_T)*>
<!ELEMENT ROW_T (a1, a2, .... An)>
<!ELEMENT a1 ....>
<!ELEMENT a2 ....>
<!ELEMENT a3 ....>
....
<!ELEMENT an ....>
```

In the exploration step 12, standardized database functions may be used to retrieve information on a database's data scheme. For example, the industry standard Open Database Connectivity ("ODBC") supports functions SQL Columns( ) allowing retrieval on the names and types of a table's attributes (columns).

Thus, it is possible to extract sufficient information from a database that allows the automatic generation of an XML DTD as outlined above and which is independent of the actual data stored in the database.

Referring now to FIGs. **1** and **2**, when the exploration step 12 identifies ASN.1 formatted data, and the associated ASN.1 meta data specification, a conversion mapping submethod 14b is activated, which generates an XML meta data description (DTD) from ASN.1 meta data.

The mapping submethod 14b covers all ASN.1 constructs, both primitive and composite. Primitive constructs are, for example, Boolean, integer, bit string, etc. Composite constructs are combinations of primitive constructs, such as SET and SEQUENCE. Specifications for Abstract Syntax Notation One can be found in the CCITT Recommendation X.208 (1988). More

detailed information about XML can be found in the article "W3C: Extensible Markup Language ("XML") 1.0, February 1998, at http://www.w3.org/TR/REC-xml. References to such specifications are, of course, routine when dealing with computer programs or standardized constructs.

Referring now to Fig. **2**, the mapping submethod 14b performs the following steps.

In a first step 30, primitive ASN.1 types are mapped onto XML entities containing character data ("CDATA"). For each primitive ASN.1 type, we define one entity, for example <!ENTITY % BOOLEAN "(true|false)">, <!ENTITY % INTEGER "CDATA"> and so on.

In a second step 32, fields of ASN.1 constructs, which are of a primitive ASN.1 type, become XML elements with an attribute of the corresponding entity. For example, if the method 10 finds a specification of an INTEGER element in ASN.1 (e.g. a INTEGER), it generates an XML element as follows:

```
<!ELEMENT a>,
<!ATTLIST a
        value INTEGER #REQUIRED
        >
```

In a third step 34, the ASN.1 "ANY" type is mapped onto an XML entity containing un-interpreted data ("PCDATA").

In a fourth step 36, the ASN.1 constants are mapped onto XML entities.

In a fifth step 40, the ASN.1 "SEQUENCE" construct is mapped onto an XML element containing the individual elements of the SEQUENCE as an XML sequence. Optional ASN.1 elements are handled with the XML "?" operator.

In a sixth step 42, the ASN.1 "SEQUENCE OF" construct is mapped onto the XML repetition construct of base elements "*". The ASN.1 required ordering of SEQUENCE OF is implicit

through the ordering with the repetition indicated via XML's "*" operator.

In a seventh step 44, the ASN.1 "SET" construct is mapped onto the XML choice construct. Optional ASN.1 elements in a SET are handled through the "?" operator.

In an eighth step 46, the ASN.1 "SET OF" construct is mapped onto the XML repetition construct of "*".

In a ninth step 50, the ASN.1 "CHOICE" construct is mapped onto an XML element containing the individual elements of the CHOICE as XML alternatives by using the "|" operator.

In a tenth step 52, the ASN.1 "COMPONENTS OF" construct is mapped as follows:

> For every COMPONENTS OF construct, an XML element is created which contains the individual component elements.

The XML construct for the ASN.1 "COMPONENTS OF [ASN.1 Construct]" expression optionally has an attribute that refers to the contained construct. Thus, an ASN.1 construct such as

```
B:= SEQUENCE {a INTEGER, b OCTET STRING};
A:= SEQUENCE {COMPONENTS OF B, c UNIVERSAL TIME}
Becomes the XML DTD specification
<!ELEMENT ComponentsOf_B (a, b)>
<!ATTLIST ComponentsOf_B
        origin CDATA #REQUIRED
>
<!ELEMENT A (ComponentsOf_B, c)>
```

This backward reference to the component container is necessary to avoid loss of information

contained in the ASN.1 meta-data description.

In an eleventh step 54, each ASN.1 module is mapped onto a specific XML DTD by recursively applying the above rules to the module's contents. Each DTD forms a separate name space. The ASN.1 constructs IMPORT and EXPORT can then be modeled using qualified names for imported/exported XML elements.

In order to perform a lossless mapping (in that no information is lost in the mapping) onto XML, the submethod 14b can insert ASN.1 tags into the XML specification by including either TAG elements or three TAG attributes in the XML specification. XML "TAG" elements use an attribute to indicate their policy, i.e., implicit or explicit, their nature, i.e., universal, application-wide, context-specific, or private-use and the tag's value. XML tag attributes in a separate name-space asn are ASN:policy (EXPLICIT or IMPLICIT), ASN:class (UNIVERSAL, APPLICATION, CONTEXT, PRIVATE), and ASN:tag containing the original ASN.1 tag value. Note that the preservation of tagging information is only necessary for a lossless mapping and can therefore be considered optional.

An important assumption that has been made until now is that names are unique. This can be achieved by flattening the nested name structure of any ASN.1 construct through concatenating the ASN.1 names. For example, B:=SEQUENCE {a INTEGER, b OCTET STRING} would generate XML element names B, B_a, and B_b.

The above handling of COMPONENTS OF guarantees lossless mapping. However, ASN.1 encoders ignore that information and thus it is conceivable to also omit this information when defining the XML mapping. The above example would simply become

<!ELEMENT A (B_a, B_b, c)>

Multiple variations and modifications are possible in the embodiments of the invention described here. Although certain illustrative embodiments of the invention have been shown and described here, a wide range of modifications, changes, and substitutions is contemplated in the foregoing disclosure. In some instances, some features of the present invention may be

employed without a corresponding use of the other features. Accordingly, it is appropriate that the foregoing description be construed broadly and understood as being given by way of illustration and example only, the spirit and scope of the invention being limited only by the appended claims.